Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»
Институт математики, информатики и информационных технологий
Кафедра информатики, информационных технологий
и методики обучения информатике

ТЕХНОЛОГИЯ РАЗРАБОТКИ ИГРОВЫХ ПРИЛОЖЕНИЙ ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТА UNITY3D

Выпускная квалификационная работа
по направлению подготовки 09.03.02 - Информационные системы и технологии

Работа допущена к защите
«____»_____ 2016 г.
Зав. кафедрой _____

Исполнитель: студентка группы ИС-41
ИМИиИТ
Киркор М.А.
Руководитель: к.п.н, доцент кафедры
ИИТиМОИ
Газейкина А.И.

Екатеринбург – 2016

Реферат

Киркор М.А. ТЕХНОЛОГИЯ РАЗРАБОТКИ ИГРОВЫХ ПРИЛОЖЕНИЙ ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТА UNITY3D, выпускная квалификационная работа: 60 стр., рис. 26, табл. 4, библ. 19 назв., приложений 2.

Ключевые слова: операционная система Android, игровой движок (game engine) Unity3D, мобильное приложение, инструменты разработки.

Объект исследования: процесс разработки игровых приложений для операционной системы Android.

Цель работы: создать технологию разработки игровых приложений и игровое приложение под управлением операционной системы Android при помощи инструмента Unity3D.

Работа направлена на разработку игрового мобильного приложения по созданной технологии разработки игр под операционную систему Android посредством Unity3D. Рассматривается специфика разработки приложений на платформе Android, исследуются инструменты и сервисы для разработки Android-приложений. Производится анализ инструмента Unity3D.

В ходе работы было создано мобильное приложение «Потерянная шляпа» для операционной системы Android с использованием инструмента Unity3D. Исходные коды написаны на языке C#.

Созданное мобильное приложение прошло апробацию в Уральском государственном педагогическом университете.

Оглавление

Введение

В XXI век – век информационного общества, человек получает очень большое количество информации, которую необходимо проанализировать и структурировать. Для дальнейшего использования полученной информации необходимо обеспечить ее хранение. На сегодняшний день данную задачу помогают решить мобильные устройства, такие как: смартфоны, планшеты, электронные книги, нетбуки и т.д. Однако вышеописанные переносные устройства используются не только для работы, но и для деятельности развлекательного характера. Из-за большой информационной нагрузки появилась потребность в частой смене деятельности. На помощь в решении данной проблемы пришли мобильные игровые приложения, не несущие тяжелой смысловой нагрузки, тем самым способствуя отдыху человека и переключению внимания с одной деятельности на другую. Исходя из вышеизложенного следует, что игры являются одним из основных продуктов на рынке информационных продуктов и услуг, что позволяет разработчикам расширить круг производимых приложений. Также следует отметить, что игровые приложения привносят большой вклад в развитие информационных технологий, например, в развитие программного и аппаратного обеспечения. Благодаря растущим требованиям пользователя к качеству игр увеличиваются требования к производительности устройств, что в свою очередь влечет за собой разработку новых технологий. Именно поэтому разработка игровых приложений на сегодняшний день является наиболее востребованной среди разработчиков мобильных приложений.

В связи с большим количеством устройств, работающих под операционной системой Android, разработка приложений под данную платформу является актуальной.

Объект исследования: процесс разработки игровых приложений для операционной системы Android.

Предмет исследования: технология разработки игровых приложений для операционной системы Android при помощи инструмента Unity3D.

Цель исследования: создать технологию разработки игровых приложений и игровое приложение под управлением операционной системы Android при помощи инструмента Unity3D.

Задачи:

1. Проанализировать особенности разработки приложений для различных мобильных платформ, таких как: Windows Phone, iOS, Android.
2. Рассмотреть возможности инструмента Unity3D.
3. Обосновать технологию разработки игровых приложений.
4. Разработать игровое приложение «Потерянная шляпа» средствами Unity3D для операционной системы Android.
5. Опубликовать разработанное приложение в Google Play и собрать статистические данные.

Глава 1. Теоретические основы разработки приложений для мобильных платформ

1.1 Разработка приложений для мобильных платформ

Разработка приложений для мобильных платформ – это процесс, в результате которого создаются приложения для переносных устройств, таких как смартфоны или планшетные компьютеры. Данные приложения могут быть предустановленны на устройство в процессе производства, загружены пользователем с помощью различных источников распространения программного обеспечения или являться веб-приложениями, которые обрабатываются на стороне клиента или сервера.

Разработку приложений необходимо начинать с выбора подходящей платформы, для которой будет создаваться мобильное приложение. Существует несколько понятий мобильной платформы, однако в данном контексте уместно считать, что это операционная система для мобильных устройств. Наиболее востребованными и актуальными на сегодняшний день принято считать следующие платформы:

- Windows Phone – мобильная платформа, разработанная американской компанией Microsoft;
- iOS – мобильная платформа для смартфонов, электронных планшетов и носимых проигрывателей, разработанная американской компанией Apple;
- Android – мобильная платформа для смартфонов и множества других устройств, разработанная американской компанией Google.

Рассмотрим описанные выше платформы более подробно, выделим их особенности и определим наиболее востребованную операционную систему.

Windows Phone. По статистике за период с апреля по июнь 2015 года Windows Phone находится на третьем месте среди наиболее популярных мобильных операционных систем и занимает лишь 2,5% доли рынка [15].

Главным отличием данной платформы являются: возможность работы приложений в фоновом режиме, а также самостоятельный контроль системы за количеством оперативной памяти. Еще одной отличительной особенностью является то, что у данной системы есть ограничения на время запуска приложений, а также на размер загружаемых файлов, который зависит от интернет соединения, количества фоновых операций и т.д. Именно поэтому приложения не всегда корректно работают [16].

Разработка приложений для Windows Phone ведется на следующих языках: C# и Visual Basic .NET.

Приложения для данной платформы размещаются в Windows Phone Store. Windows Phone Store – это магазин приложений Microsoft для мобильной платформы Windows Phone, который позволяет пользователям устанавливать и приобретать различные приложения и игры [14]. Информация о размещении готовых приложений в магазине приведена в Приложении 1.

iOS. В свою очередь, данная мобильная операционная система занимает второе место, и ее доля на рынке составляет 14,4% [15]. Оснащена системой безопасности, которая не позволяет запускаться подозрительным приложениям, что позволяет защитить систему [6].

Разработка приложений для iOS ведется на следующих языках: Objective C и Swift.

Приложения для данной платформы размещаются в App Store. App Store – магазин приложений, раздел онлайн-магазина iTunes Store, содержащий различные приложения для мобильных телефонов iPhone, плееров iPod Touch и планшетов iPad, а также для персональных компьютеров Mac и позволяющий их купить, либо скачать бесплатно. Информация о размещении готовых приложений в магазине приведена в Приложении 1.

Android. На сегодняшний день Android является лидирующей мобильной операционной системой и занимает 82% на рынке мобильных устройств [15].

Разработка приложений под операционную систему Android имеет ряд особенностей. Приложения в Android используют четыре основных типа компонентов: деятельности, сервисы, слушатели сообщений, поставщики содержимого [3].

1. Деятельность (Activity) – это визуальный компонент приложения, который отвечает за интерфейс и это то, с чем можно взаимодействовать. Приложение может содержать как несколько деятельностей, так и вовсе не содержать их.
2. Сервисы (Services) – это то что выполняется, пока приложение не находится в фокусе. Сервис может запускаться вместе с системой и работать в фоновом режиме (например, музыкальный проигрыватель), выполняя некоторые действия. Взаимодействовать с сервисом можно посредством интерфейсов.
3. Слушатель событий (Listener) – один из важнейших компонентов приложения. Слушатель событий, также как и сервис, не имеет видимого интерфейса. Его задача отслеживать определенные действия или системные сообщения и реагировать на это. Так как сам по себе слушатель событий сделать ничего не может, он передает сигнал дальше (пользователю приходит уведомление, по событию вызывается деятельность и т.д.).
4. Поставщик содержимого (Content provider) предоставляет определенные данные другим приложениям. Эти данные могут храниться в файловой системе, SQLite базе данных и т.д.

В отличие от домашних компьютеров в Android нет понятия "завершить работу приложения". Система держит программу в памяти, пока эта память ей не потребуется для других целей. Метод finish() класса деятельности или stopService() класса сервиса завершает работу компонента (деятельности или сервиса), но не завершает программу. Так, например, если в этих компонентах были запущены фоновые нити, то они продолжают работу.

Одним из уникальных качеств Android является то, что все программы имеют общий уровень. Android не проводит различия между основными сис-

темными программами и программами сторонних разработчиков. Все они пользуются равными правами доступа к возможностям мобильного устройства, предоставляя пользователям широкий спектр приложений и услуг. С устройствами, построенными на платформе Android, пользователи будут иметь возможность в полной мере адаптировать устройство под свои интересы.

Отрицательной особенностью операционной системы Android является то, что для установки приложения размером N Мб, нужно от 2N до 4N Мб свободного места [2]. Это происходит из-за того, что сначала файл программы загружается на устройство и занимает там N Мб места, потом он будет распакован для проверки, где займет от N до 2N Мб в среднем, затем он будет перемещен на свое место в файловой системе, что потребует еще N Мб.

Особенности реализации файловой системы операционной системы Android. Чем меньше свободного места на SD-карте, тем больше времени занимают операции с файлами. Время доступа может вырасти до 200 раз. Это происходит из-за того что файловая система отслеживает интенсивные операции с файлами и наиболее востребованные файлы пытается перемещать с места на место. Это сделано для того, что бы равномерно распределить нагрузку (чтение и запись) по всей SD-карте. Чем больше файлов и чем меньше свободного места, тем больше ей на это нужно времени. Проблема решается путем замены множества небольших файлов одним большим.

Также существует перечень особенностей, которые необходимо знать для разработки приложений под операционную систему Android [4]:

- различные варианты разрешения экрана. Разрешение экрана – это размеры получаемого на экране изображения в пикселях. Разрешение экрана устройств может варьироваться от 320x240 пикселей до 1920x1080 и выше;
- использование множества аппаратных конфигураций. Данная особенность выражается в производительности различных устройств, кото-

рая зависит от процессора, оперативной памяти, видеоускорителя и т.д.;

- существование множества вариантов программного обеспечения. Производители устройств могут изменять по своему усмотрению операционную систему Android благодаря открытому исходному коду;
- создание исходной графики в HD формате. Графика высокого разрешения адаптируется под различные разрешения экрана;
- поддержка формата текстур и звуков зависит от конкретного устройства.

На текущий момент разработка Android-приложений является быстроразвивающимся и перспективным направлением.

Разработка приложений под Android ведется на следующих языках: Java, Delphi, HTML5, JavaScript, Python, C# и C/C++ и т.д.

Для работы с Android-приложениями используют виртуальные машины [13]. Среда выполнения приложений представляет собой набор инструкций, выполняемых для перевода кода приложения в код, понятный компьютеру. Виртуальная машина задействуется, даже не являясь частью основной программы. Все компьютерные языки нуждаются в среде выполнения для запуска приложений, написанных на них.

Использование виртуальной машины имеет ряд преимуществ:

- Безопасность – выполнение кода стороннего приложения происходит в изолированной среде. Следовательно, потенциально вредоносный код не сможет навредить ядру операционной системы.
- Совместимость – приложения, написанные под Android, содержат нескомпилированные инструкции, которые компилируются виртуальной машиной непосредственно перед началом управления. Это, в свою очередь, значительно облегчает процесс разработки программ.

Приложения для Android представляют собой программы для виртуальной машины Dalvik [1]. Виртуальная машина Dalvik проектировалась специ-

ально под платформу Android и впервые нашла применение в Android 1.0 в 2008 году, и использовалась до выхода Android 5.0 Lollipop. При разработке программ под Android исходный код программы переводится компилятором в специальный машинно-независимый низкоуровневый код. На выходе получается исполняемый файл Android Package (APK).

Принцип работы Dalvik virtual machine заключается в том, что система компилирует приложения во время их запуска. Суть данного метода состоит в том, что перевод кода программы на машинный язык осуществляется каждый раз, когда пользователь запускает приложение. Однако такое решение отрицательно сказывается на производительности из-за высокой нагрузки на процессор во время компиляции. Dalvik собирает часть скомпилированных данных за время использования программы, после чего отправляет их в кэш. Таким образом, в случае повторного запуска приложения, система извлекает часть полученного кода из кэша, который хранится в файловой системе, что положительно сказывается на производительности. Однако если пользователь задействует новые функции приложения, неиспользованные им ранее, происходит повторное преобразование кода.

Виртуальная машина ART, пришедшая на замену Dalvik, впервые появляется в Android 4.4 KitKat. В 2014 году среда выполнения ART еще не использовалась в качестве основной и находилась на экспериментальной стадии. Полномасштабный переход с Dalvik на ART был осуществлен в Android 5.0 Lollipop.

Главным преимуществом виртуальной машины Android Runtime является новый тип компиляции. Суть данного метода состоит в том, что система компилирует приложения до его запуска, во время установки. То есть, пользователь запускает программу уже с преобразованным кодом, что значительно ускоряет процесс ее загрузки, прежде всего из-за малого количества потребляемой оперативной памяти.

Глобальным изменениям подвергся системный сборщик мусора, из-за которого отображение интерфейса периодически происходит с задержкой. На данный момент время, выделенное на сборку мусора, практически равно нулю: вместо десятков миллисекунд работа сборщика мусора выполняется в течении 1 миллисекунды.

Еще одной особенностью виртуальной машины ART является поддержка 64-битных процессоров и приложений, которая отсутствует в Dalvik virtual machine. Однако к недостаткам новой среды выполнения можно отнести больший объем памяти, требуемый для установки приложений, а также более длительный процесс установки.

Таблица 1.
Главные отличия виртуальных машин

Dalvik virtual machine	*Android Runtime (ART)*
Используется меньше постоянной памяти, однако дольше запускаются приложения	Задействуется больше постоянной памяти, однако быстрее открываются программы и меньше используется процессор
Загрузка устройства происходит быстрее, так как кеш приложений создается постепенно	Загрузка устройства происходит медленнее, так как кэш всех приложений создается при включении аппарата
Больше подходит для устройств с маленьким объемом постоянной памяти	Занимает больше постоянной памяти, так как хранит уже скомпилированное приложение вместе с .APK
32 bit	32 bit 64 bit

Таким образом, каждая из виртуальных машин имеет свои достоинства, которые влияют на ее выбор.

Для распространения созданных приложений существует множество доступных механизмов, например, магазины приложений, такие как: Google Play, Amazon, Yandex Store и множество других. Так же возможно распространение игры в интернете в свободном доступе.

Google Play – это магазин приложений, игр, книг, музыки и фильмов компании Google и других компаний, позволяющий владельцам устройств с операционной системой Android устанавливать и приобретать различные приложе-

ния. Информация о размещении готовых приложений в магазине приведена в Приложении 1.

Рассмотрев и проанализировав три наиболее популярные платформы для разработки мобильных приложений, можно заключить, что Android является наиболее подходящей операционной системой для создания приложений по следующим причинам:

- занимает первое место по распространенности на рынке мобильных устройств;
- имеет открытый исходный код;
- позволяет скачивать приложения из других источников, помимо официального магазина приложений;
- аккаунт разработчика имеет самую низкую стоимость;

В связи с тем, что для создания мобильного приложения была выбрана операционная система Android, были рассмотрены средства разработки, которые позволяют создавать приложения для данной платформы.

1. Qt [10] – кроссплатформенный инструментарий разработки программного обеспечения на языке программирования C++. Существует возможность подключения других языков программирования (Python, Ruby, Java, PHP и т.д.). Разработка приложений ведется для следующих платформ: Android, IOS, Windows, Linux / X11, OS X, Windows Runtime, WinCE;
2. Construct 2 [11] – конструктор двумерных игр разрабатываемых на языке JavaScript. Разработка приложений ведется для следующих платформ: PC, Mac, Linux, браузеры с поддержкой HTML5, Android, iOS, Windows Phone, Facebook и другие;

   Достоинства:

   - возможность создания игр под iOS и Android без программирования;
   - кроссплатформенность;
   - возможность создания мультиплеерных 2D игр с минимальными ограничениями

- возможность установки сторонних плагинов;
- прост в освоении;

Недостатки:

- отсутствует возможность создания 3D игр;
- в бесплатной версии отсутствуют множество функций, звуковое сопровождение, эффекты, действует лимит спрайтов;

3. Game Maker: Studio [12] – программа для создания компьютерных игр на скриптовом языке GML. Разработка приложений ведется для следующих платформ: Windows, Mac OS X, Ubuntu, Android, iOS, Windows Phone, Tizen, Xbox, PlayStation;

   Достоинства:

   - кроссплатформенность;
   - поддержка библиотек и расширений, в том числе на разных языках;
   - гибкая ценовая категория, Standard версия Game Maker: Studio является бесплатной;
   - собственный упрощенный язык программирования Game Maker Language (GML);
   - интеграция с несколькими системами управления версиями;
   - интеграция со Steam API;

   Недостатки:

   - несмотря на возможность работы с 3D, в Game Maker она крайне неудобна;
   - сама среда разработки Game Maker: Studio доступна только на Windows.

4. Unity3D [9] – это кроссплатформенный инструментарий для разработки двух- и трехмерных приложений и игр на JavaScript или C#. Разработка приложений ведется для следующих платформ: Windows, MacOS, Wii, Apple iOS, Android, PS3 и XBox 360.

Достоинства:

- кроссплатформенность;
- хорошо проработанный физический движок;
- возможность создания 2D и 3D приложений;
- разработка ведется на двух языках Java Script и C#;
- прост в освоении;

Недостатки:

- закрытый код;
- невозможность вносить изменения в физическое ядро и дополнять его сторонними возможностями (например, нельзя изменять модель физического взаимодействия объектов).

Таблица 2.
Анализ средств разработки

Средства разработки	*Встроенная физика*	*Графика*	*Кроссплатформенность*	*Языки программирования*	*Игровая направленность*	*Бесплатная лицензия для коммерческого использования*
Qt	-	2D	+	C++	-	-
Construct 2	+	2D	+	Java Script	+	+
Game Macer: Studio+	+	3D	+	GML	+	-
Unity3D	+	3D	+	Java Script, C#	+	+

Проанализировав средства разработки, было выявлено, что Qt не специализирован на создании игровых приложений. Оставшиеся три средства очень схожи и имеют незначительные недостатки. Однако для выпускной квалификационной работы был выбран Unity3 по ряду следующих причин:

- программирование на C#;
- бесплатная лицензия для коммерческих разработок;
- возможность создавать 3D модели;

- наличие магазина Asset Store, который предоставляет бесплатные материалы для помощи в разработке приложений.

1.2 Анализ возможностей Unity3D

Unity3D, разработанный компанией Unity Technologies, представляет собой кроссплатформенный игровой движок для разработки двух- и трехмерных приложений со встроенной технологией IDE и является одним из наиболее популярных игровых движков для платформы Android. Он применяется для разработки видеоигр для веб-платформ, платформ настольных ПК, игровых консолей и мобильных устройств, и используется более чем миллионом разработчиков. Разработка приложений ведется для следующих платформ: Windows, MacOS, Wii, Apple iOS, Android, PS3 и XBox 360. Для Unity3D предусмотрено несколько лицензий: Unity 5 Personal Edition (бесплатная), Unity 5 Professional Edition, iOS Pro, Android Pro и групповая лицензия Team License.

Рисунок 1. Изображение иконок поддерживаемых платформ

Факты о компании:

- Unity3D занимает более чем 45% мирового рынка полнофункциональных игровых движков, что приблизительно в 3 раза больше, чем у ближайшего конкурента.

Рисунок 2. Диаграмма распространенности игровых движков

- Игровой движок Unity3D гораздо популярнее среди разработчиков, чем любое другое программное обеспечение. Также все время растет доля разработчиков, опирающихся на Unity3D как на первичный инструмент разработки и использующих его. Среди клиентов Unity3D есть: Cartoon Network, Coca-Cola, Disney, Electronic Arts, LEGO, Microsoft, NASA, Nexon, Nickelodeon, Square, Ubisoft и Warner Bros. Крупные и маленькие студии, независимые профессионалы, и все больше разработчиков переходят на Unity3D.

Рисунок 3. Диаграмма статистики использования различных инструментов

- На любом крупном рынке в мире подавляющее большинство топовых мобильных 3D-игр, сделанных с помощью сторонних инструментов, созданы с использованием Unity3D.

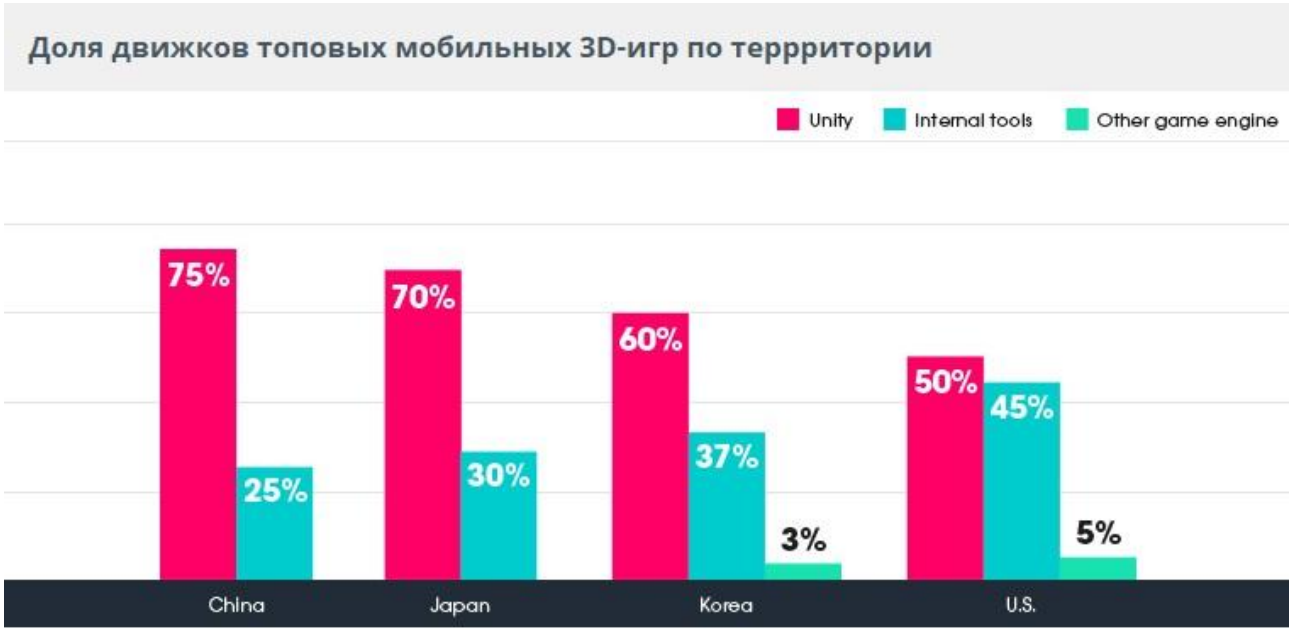


Рисунок 4. Диаграмма статистики использования Unity3D в разных странах

Прежде чем начать более детальный анализ инструмента Unity3D выделим его особенности, которые также являются причиной выбора данного движка для реализации игрового приложения [7]:

- кроссплатформенность;

- наличие бесплатной версии с некоторыми ограничениями. Также стоит отметить, что распространение игр, созданных в бесплатной версии, возможно при условии, что ежегодный доход не превышает $100000.
- возможность создания 2D и 3D приложений;
- возможность писать на двух языках программирования: JavaScript, C#;
- возможность создания игры любого жанра;
- возможность мгновенного запуска игры через встроенный эмулятор;
- IDE: сочетание редактора сцен (в комплексе общего редактора) с редактором игровых объектов и редактора скриптов;
- интуитивно понятный и полностью настраиваемый интерфейс;
- работа с ресурсами через Drag-and-Drop. Drag-and-Drop (в пер. с англ. тащи и бросай) – это способ оперирования элементами интерфейса при помощи манипулятора «мышь» или сенсорного экрана, который позволяет захватить элемент и перенести его;
- наличие магазина Asset Store, в котором можно найти: 3D модели, расширения редактора, скрипты, шейдеры (компьютерные программы, предназначенные для исполнения процессорами видеокарты), звуковые файлы, анимацию и многое другое.

UNITY 5 Что включено	PERSONAL EDITION	PROFESSIONAL EDITION
Движок со всеми функциями	✓	✓
Без авторских отчислений	✓	✓
Все платформы (применимы ограничения)	✓	✓
Изменяемый заставочный экран	✗	✓
Unity Cloud Build Pro - 12 месяцев	✗	✓
Unity Analytics Pro	✗	✓
Team License	✗	✓
Приоритетная обработка ошибок	✗	✓
Game Performance Reporting	✗	✓
Доступ к бета-версиям	✗	✓
Неограниченный доход и бюджетирование	✗	✓
Включены будущие платформы	✗	✓
Профессиональная обложка редактора	✗	✓
Asset Store Level 11	✗	✓
Возможности сообщества для пользователей Pro	✗	✓
Доступ к исходному коду	✗	$
Premium support	$	$
	БЕСПЛАТНАЯ ЗАГРУЗКА	ОТ $75/МЕСЯЦ

Рисунок 5. Таблица сравнения лицензий

Сцена. Можно считать каждый файл сцены отдельным игровым уровнем. В каждой сцене можно разместить объекты окружения, заграждения, декорации, последовательно создавая дизайн и саму игру.

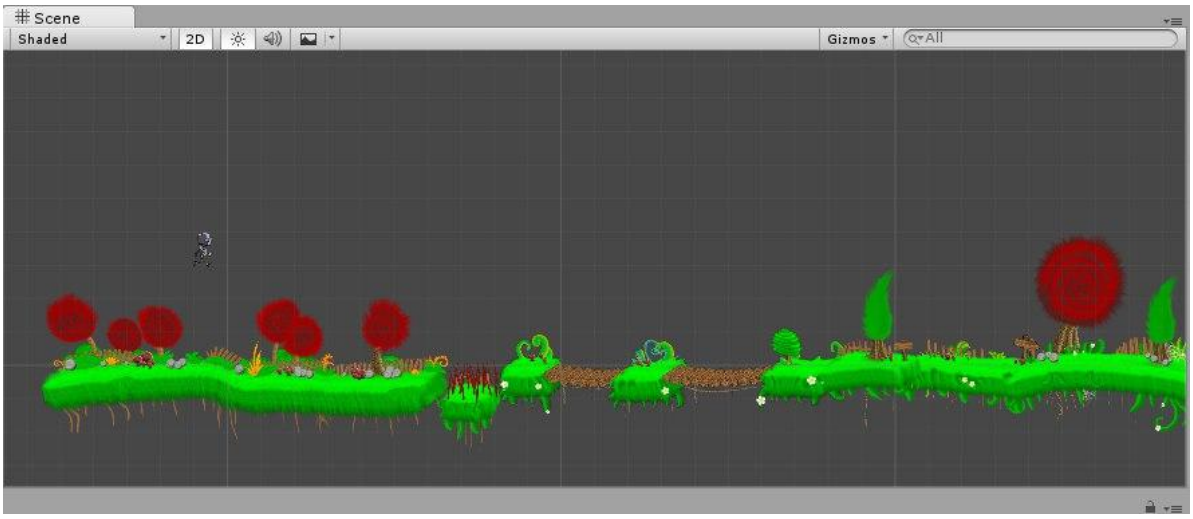


Рисунок 6. Изображение окна сцены

Игровой объект. Он является самой важной частью сцены, представляет собой контейнер, который может содержать в себе различные элементы. Сам по себе игровой объект ничего не может сделать. Он нуждается в настройке для того, чтобы стать игровым персонажем, интерактивным объектом, с которым можно взаимодействовать, или частью статического окружающего мира. В свою очередь каждый игровой объект составляется из компонентов. В зависимости от того, что игровой объект должен делать, к нему добавляются различные комбинации компонентов.

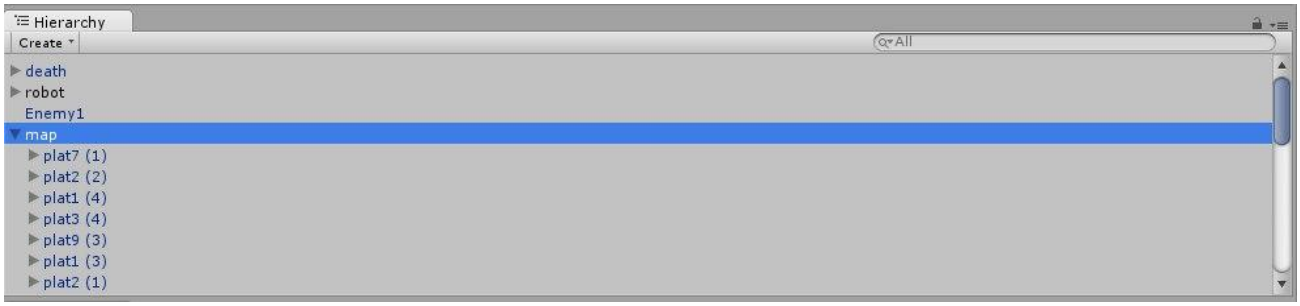


Рисунок 7. Изображение списка объектов, расположенных на сцене

Компоненты. Они являются функциональной частью каждого игрового объекта и определяют его поведение в игре. Самый важный компонент – Transform, который есть как у видимых, так и у невидимых объектов. Без данного компонента невозможно создать ни один игровой объект, поэтому он всегда установлен по умолчанию и определяет положение, вращение и масштаб игрового объекта в игровом мире. Помимо компонента Transform существует множество других, которые добавляются к игровому объекту при необходимости.

Все компоненты можно поделить на несколько видов:

1. Компоненты для моделирования физических тел и процессов. Если к игровому объекту добавить данный компонент, то с ним можно будет обращаться как с физическим телом, а также на него будут действовать законы физики.

2. Компоненты для визуального отображения объектов. Могут представлять игровые объекты как 3D моделью с различными текстурами, так и 2D изображениями.
3. Аудио компоненты. Данные компоненты предназначены для работы со звуками.
4. Скрипты. Программный код, который описывает поведение игровых объектов.

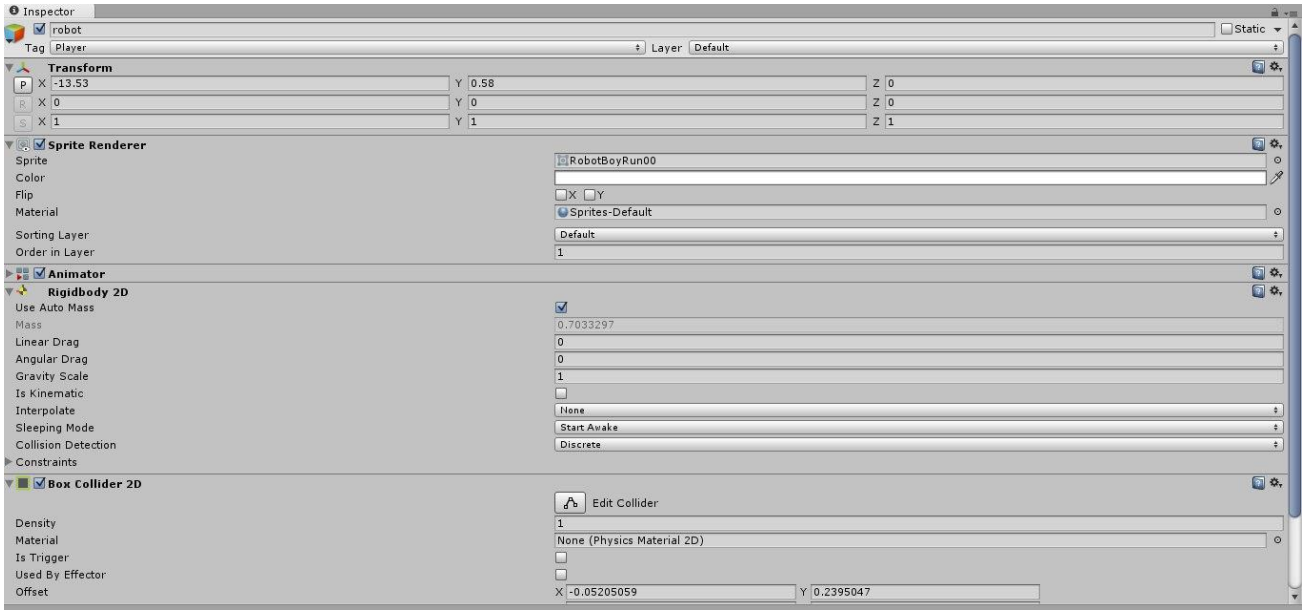


Рисунок 8. Изображение списка компонентов

Скрипты. Скрипты нужны для того, чтобы создавать игровую логику и описывать поведение игровых объектов. Они позволяют активировать игровые события, изменять параметры компонентов, и реагировать на действия пользователя. Скрипт взаимодействует с внутренними механизмами Unity3D за счет создания класса, наследованного от встроенного класса, называемого MonoBehaviour. Существует две основные функции, определенные внутри класса. Функция Update – это место для размещения кода, который будет обрабатывать обновление кадра для игрового объекта, например: движение, обработка событий и ответная реакция на ввод пользователя, в основном всё, что должно быть обработано с течением времени в игровом процессе. Функция Start будет вызвана Unity до начала игрового процесса, то есть до первого вызо-

ва функции Update. Скрипт начнет работать после присоединения к игровому объекту и нажатия на кнопку Play.

По умолчанию для написания скриптов используется IDE MonoDevelop. Так же есть возможность использовать другие IDE, например, Vissual Studio от компании Microsoft. MonoDevelop – является средой разработки, предназначенной для создания приложений на следующих языках: CIL, C#, Nemerle, Java, Boo, Vala, Visual Basic .NET, C и C++.

Возможности среды:

- встроенный отладчик;
- сворачивание кода;
- автодополнение кода;
- браузер классов;
- подсветка синтаксиса;
- модульное тестирование;
- поддержка плагинов.

Скрипты для Unity3D можно писать на нескольких языках программирования:

- C# – объектно-ориентированный язык программирования;
- JavaScript – прототипно-ориентированный сценарный язык программирования.

При создании нового скрипта содержимое файла будет выглядеть следующим образом:

```
using UnityEngine;
using System.Collections;
public class MainPlayer : MonoBehaviour {
 // Use this for initialization
 void Start () {
 }
```

```
// Update is called once per frame
void Update () {
}
}
```

Ресурсы. Это составные блоки всех проектов Unity3D. Игровой движок ссылается на файлы изображений, 3D моделей, звуков и т.д., которые будут использоваться, при создании игры, в качестве ресурсов.

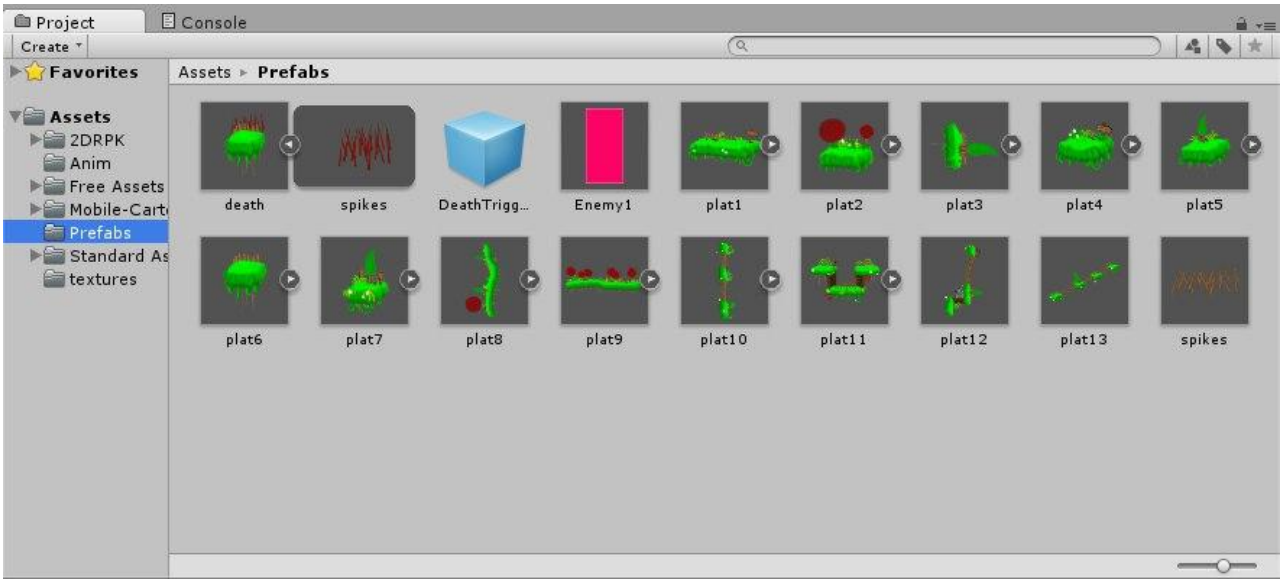


Рисунок 9. Изображение ресурсов

Графика и звук. Для любой игры необходимы графическое и звуковое оформление. В среде разработки есть целый ряд понятий и связанных с ними технологий, которые можно отнести к технологиям по работе с графикой:

- меш;
- материал;
- текстура;
- шейдер.

Меш – это координатная сетка, состоящая из треугольников. Она задается массивом из треугольников, треугольник в свою очередь задается смещением каждой вершины по материалу. В итоге при необходимости можно получить нелинейное отображение текстуры. Меш бывает как полигональный, так и состоящий из двух треугольников.

Материал – это структура, которая содержит разнообразные параметры по отображению текстуры, такие как: способ отображения, уровень фильтрации, а также смещение и масштаб на текстуре. Материал и меш в Unity3D существуют раздельно, так как меш может отвечать за физику объекта. Материал может быть как общим, т.е. принадлежать сразу нескольким объектам, так и частным.

Текстура – это массив цветов каждого пикселя. В Unity3D есть класс, который представляет текстуру – это Texture2D. Он содержит набор полей и методов непосредственно для работы с текстурой, такие как: «изменить пиксель» или «применить компрессию».

Шейдер – это подпрограмма по отрисовке текстуры. Обычно она пишется на собственном языке. И концепция написания строится по следующему принципу: задается последовательность отрисовки и на каждом этапе применяются фильтры или алгоритмы. Шейдер может быть рассчитан на несколько текстур.

Работа со звуком в Unity3D относительно проста, не смотря на наличие больших возможностей. Можно накладывать фильтры, закреплять звуковые эффекты за объектами, в результате чего громкость и стереоэффект регулируется автоматически. Для того чтобы закрепить за объектом звук, необходимо создать и присоединить к объекту компонент AudioSource, а потом, обращаясь к методам, добавлять к проигрыванию необходимые ресурсы.

Физика. Встроенные в Unity3D физические движки обеспечивают компонентами для обработки симуляции физики. В Unity3D существует два отдельных физических движка, один для 3D физики и один для 2D физики. Основные понятия идентичны в обоих движках (за исключением дополнительного измерения в 3D), но они реализованы с разными компонентами.

Твердые тела (Rigidbody) – это основной компонент, подключающий физическое поведение для объекта. С прикреплённым Rigidbody, объект немедленно начнёт реагировать на гравитацию. Если добавлен один или несколько

компонентов Collider, то при коллизиях (столкновениях) объект будет передвигаться.

Коллайдеры (Colliders). Коллайдер компоненты определяют форму объекта, по которой будут определяться физические столкновения. Коллайдер является невидимым и не обязательно должны быть точно такой же формы, как видимая часть объекта.

Простейшими являются так называемые примитивные типы коллайдеров. В 3D это Box Collider , Sphere Collider и Capsule Collider . В 2D это Box Collider 2D и Circle Collider 2D. Они могут быть добавлены к одному объекту в любом количестве, чтобы создать сложные коллайдеры.

Сочленения (Joints) – с помощью данного компонента можно добавить один rigidbody объект к другому или к фиксированной точке в пространстве. Unity предоставляет различные Joint компоненты, реализующие разные ограничения. Например, Hinge Joint позволяет осуществлять вращение вокруг заданной точки и оси, в то время как Spring Joint держит объекты раздельно, но расстояние между ними может немного пружинить. Как обычно, названия 2D компонентов оканчиваются на 2D, например, Hinge Joint 2D.

Контроллеры персонажа (Character Controllers). Персонажу в игре от первого или третьего лица часто требуется некоторая физика, основанная на столкновениях, чтобы он не падал сквозь пол или не ходил сквозь стены. Обычно, ускорение и движение персонажа физически нереалистичны, таким образом, он может ускоряться, тормозить и менять направление мгновенно, не подвергаясь влиянию импульса.

В 3D физике этот тип поведения можно создать, используя Character Controller. Этот компонент даёт персонажу простой коллайдер в форме капсулы, который всегда находится в вертикальном положении. У Controller есть свои особые функции для назначения скорости и направления объекта, но, в отличие от настоящих коллайдеров, он не требует Rigidbody и его эффекты импульса нереалистичны.

Character Controller не может проходить сквозь Static коллайдеры в сцене, поэтому он будет двигаться по полу и блокироваться стенами. Во время движения он может отталкивать Rigidbody объекты, но входящие столкновения не будут влиять на его ускорение. Это значит, что можно использовать стандартные 3D коллайдеры, чтобы создать сцену, по которой будет ходить Controller, но реалистичность не ограничивается, с точки зрения физики, поведением самого персонажа.

1.3 Техническое задание на разработку игрового приложения

1. Общие сведения.
   1.1. Наименование огранизации-заказчика: УрГПУ ИМИиИТ.
   1.2. Название продукта разработки: Потерянная шляпа.
   1.3. Назначение продукта: игровое приложение.
   1.4. Плановые сроки начала и окончания работы.

Таблица 3.
Календарный план

№ п/п	*Этапы выполнения*	*Сроки выполнения этапа*	
		Начало периода	*Конец периода*
1.	Разработка требований	3.09.15	3.10.15
3.	Реализация	3.11.15	3.02.16
4.	Тестирование	4.02.16	4.03.16
5.	Ввод в действие	4.04.16	15.05.16

2. Характеристика области применения продукта.
   2.1. Процессы и структуры, в которых предполагается использование продукта разработки: предполагается публикация продукта в магазине приложений Google Play, апробация его пользователями различных категорий, сопровождение и доработка.
   2.2. Характеристика персонала:

- Разработчик. Требуется владение инструментом Unity3D, языком программирования C# или JavaScript.
- Пользователь. Требуются минимальные умения работы с операционной системой Android.

3. Требования к продукту разработки.

3.1. Аппаратные требования:

Таблица 4.
Аппаратные требования

	Минимальные	*Рекомендуемые*
Платформа	Android 4.0	Android 4.4 и выше
Частота процессора	1 ГГц	1.3 ГГц
Количество ядер	1	2
Сенсорный экран	да	да
Разрешение экрана	480x800	1280x720
Оперативная память	512 Мб	1024 Мб
Свободное место на HDD	30Мб	50Мб

3.2. Указание системного программного обеспечения:

- Операционная система Android;
- Виртуальная машина Dalvik или среда выполнения Android-приложений ART.

3.3. Указание программного обеспечения, используемого для реализации:

- Unity 5 Personal Edition;
- MonoDevelop;
- Corel Draw X7;
- Графический редактор Gimp.

4. Требования к пользовательскому интерфейсу.

4.1. Общая характеристика пользовательского интерфейса: пользователю предоставляется главное меню с возможностью начать новую игру или загрузить сохраненную и управление игровыми объектами при помощи сенсорного экрана.

4.2. Размещение информации на экране, дизайн экрана:

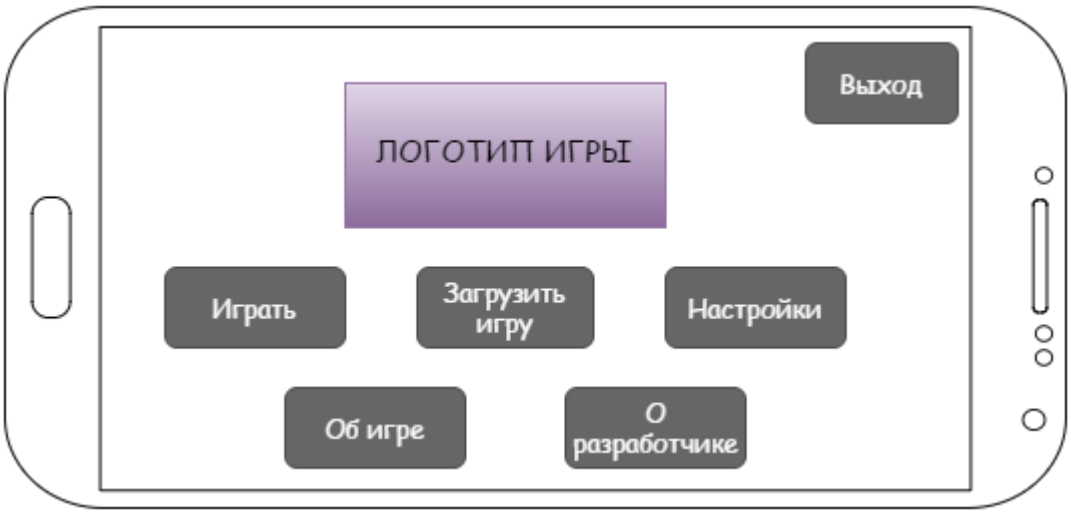


Рисунок 10. Главное меню

4.3. Особенности ввода информации пользователем, представление выходных данных: входные данные передаются с сенсорного экрана, результаты игры записываются во внешний файл.

5. Перечень сопроводительной документации.

Руководство пользователя (см. Приложение 3).

Глава 2. Разработка игрового приложения средствами инструмента Unity3D под управлением операционной системы Android

2.1 Технология разработки игрового приложения для операционной системы Android

Под технологией разработки понимают оптимальный способ ведения процесса разработки, который при определенных условиях обеспечит получение конечного продукта с заранее заданными свойствами [17].

В ходе исследования был структурирован существующий материал и на его основе создана технология разработки игровых приложений. Разработка приложения осуществляется по следующему алгоритму [8]:

1. Выбор жанра игры. Игры создаются в самых разных стилях. Различные стили игр требуют разных принципов разработки. В начале проекта необходимо выбрать жанр своей игры. Если разработчик не сможет создать что-то новое, высок шанс того, что игра будет соответствовать одному из широко распространенных жанров. Большинство жанров устанавливают стандарты игровой механики (например, схемы управления, конкретные цели и т.д.). Отклонение от этих стандартов может сделать игру успешной. Список общеизвестных жанров:

- аркадные – компьютерные игры с нарочно примитивным игровым процессом (бегает, прыгает);
- платфомер – жанр компьютерных игр, в которых основной чертой игрового процесса является перемещение по платформам и лестницам, сбор предметов, обычно необходимых для завершения уровня;
- приключение – игра, обладающая целостным сюжетом, который пишут сценаристы из области кинематографа или художественных произведений. В этом случае пользователь выступает в роли зрителя игры, и сам раскрывает все нити заранее продуманного сюжета;

- экшн – жанр компьютерных игр, в которых успех игрока в большей степени зависит от его скорости реакции и способности быстро принимать тактические решения;
- развивающие мышление и головоломки;
- казуальные игры – компьютерная игра, предназначенная для широкого круга пользователей, в которую играют от случая к случаю, чтобы как-то «убить» время. Как правило, обладает достаточно простыми правилами и не требует от пользователя хорошего владения игровым устройством;
- ролевая игра – игра, в которой пользователь управляет персонажем или группой персонажей, обладающих определенным набором навыков и умений. В процессе игры персонажи могут получать новые навыки или совершенствовать имеющиеся, за счет выполнения различных заданий;
- симуляторы – игры, которые полностью или частично имитирует определенную сферу реальной жизни. Например, имитация управления подводной лодкой, гоночным автомобилем, космическим кораблем или самолётом;
- обучающие – игры, которые обладают элементом обучающей программы. В таких играх обучение происходит через игровой процесс и способствует запоминанию полученной информации;
- спортивные игры.

Также стоит отметить, что существует несколько классификаций игр помимо вышеприведенной классификации жанров.

Классификация игр по платформам:

- персональный компьютер (ПК, PC, ноутбук, нетбук);
- игровая консоль или приставка (PS, Xbox, Nintendo);

- мобильное устройство: телефон, планшет, карманный компьютер (КПК, PDA);
- игровой автомат;
- браузерная или флеш-игра (виртуальная интернет платформа).

Классификация игр по количеству платформ:

- мультиплатформенные игры (вышедшие на двух и более платформах);
- одноплаторменные игры (эксклюзивны в рамках одной платформы).

Классификация игр по количеству игроков:

- однопользовательская;
- многопользовательская;
- многопользовательские игры на одном компьютере;
- массовые онлайновые.

2. Разработка сюжета и структуры проекта. Создание большинства игр, не зависимо от платформы, начинается с разработки сюжета. Чем подробнее описан сюжет будущей игры, тем легче построить структуру, которая включает в себя: список уровней, геймплей (игровой процесс), игровые объекты (главный герой, вражеские персонажи, логические задачи и т.д.). Первичная разработка сюжета и структуры для визуализации осуществляется "на бумаге".

3. Выбор средства разработки. Непосредственно создание приложений начинается с поиска подходящего средства разработки, например, игрового движка. Игровой движок – это ключевой модуль для игровых приложений. Средства разработки выбираются исходя из концепции игры. Существует несколько движков, работающих в среде Android, в число которых входят движки 2D и 3D на основе открытого исходного кода, а также коммерческие модули.

4. Создание рабочего прототипа. Создается концепт (графика и звуки) и разрабатывается рабочий прототип игры, в котором создаются условные объекты, временно заменяющие готовых персонажей (например: вместо персонажа используется кубик, а красивую платформу заменяет серый прямоугольник).

Когда графика создана, она "натягивается" на игровые объекты, после чего игра из прототипа переходит в демо-версию.

5. Тестирование. Прежде чем полностью создать игру, рекомендуется протестировать начальный вариант, опубликовав его или передать незаинтересованному лицу для выявления ошибок. После чего исправить ошибки и продолжить разработку игры.

6. Релиз. Протестировав приложение на работоспособность, и исправив все найденные ошибки, можно публиковать игру в магазине приложений.

Реализация представленной технологии позволяет упростить создание игрового приложения под операционную систему Android. Также предложенный алгоритм может быть использован при создании приложений для других операционных систем с поправкой на их особенности.

2.2 Разработка игрового приложения

Для создания мобильного приложения была использована созданная ранее технология для разработки игровых приложений для операционной системы Android. В качестве инструмента разработки был выбран Unity3D исходя из его достоинств, описанных в Главе 1 [5].

Для игры был выбран жанр платформер, т.к. игра предназначена для различных категорий пользователей. Разработанное приложение создано как для мобильных устройств под управлением операционной системы Android, так и для персональных компьютеров, работающих под системой Windows, из чего следует вывод, что игра является мультиплатформенной.

Перед непосредственной разработкой игрового приложения были придуманы и созданы в графическом редакторе Corel Draw X7:

- меню игры:

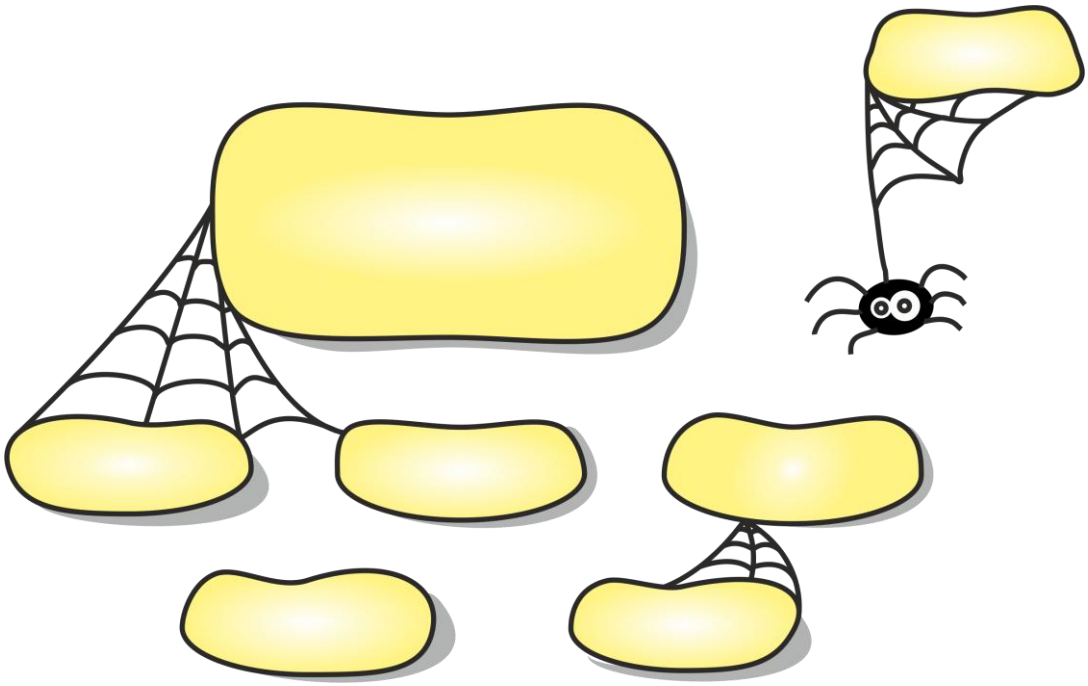


Рисунок 11. Главное меню

- персонажи игры (главный герой, враги):

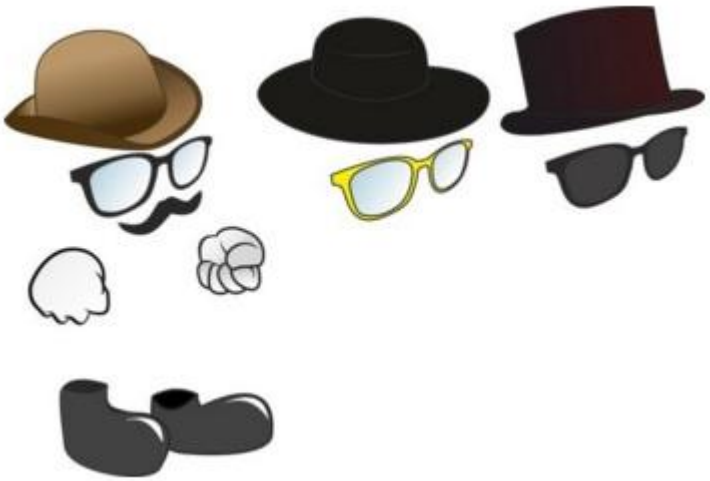


Рисунок 12. Главный герой

Рисунок 13. Отрицательные персонажи

- анимация персонажей:

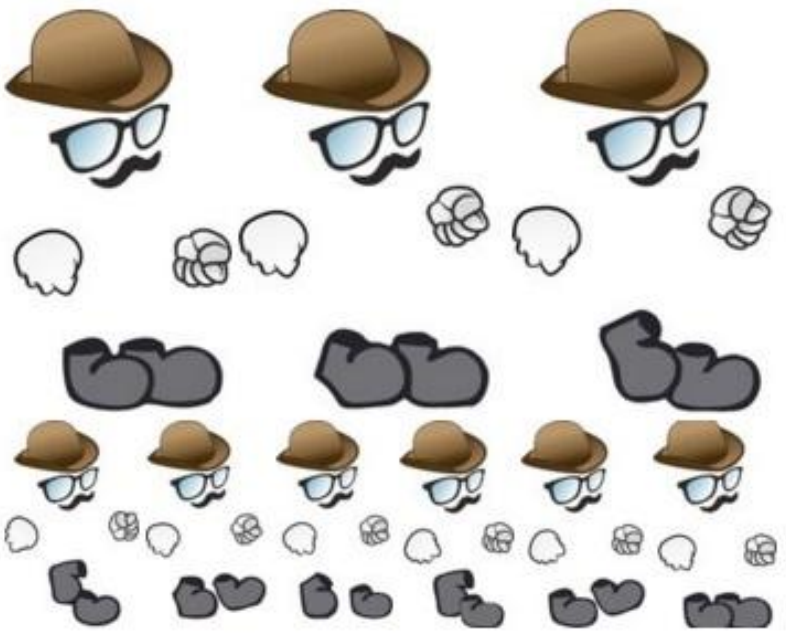


Рисунок 14. Анимация ходьбы главного героя

- дизайн игровых сцен

Рисунок 15. Фон игрового уровня

Также были заимствованы бесплатные материалы из магазина Asset Store.

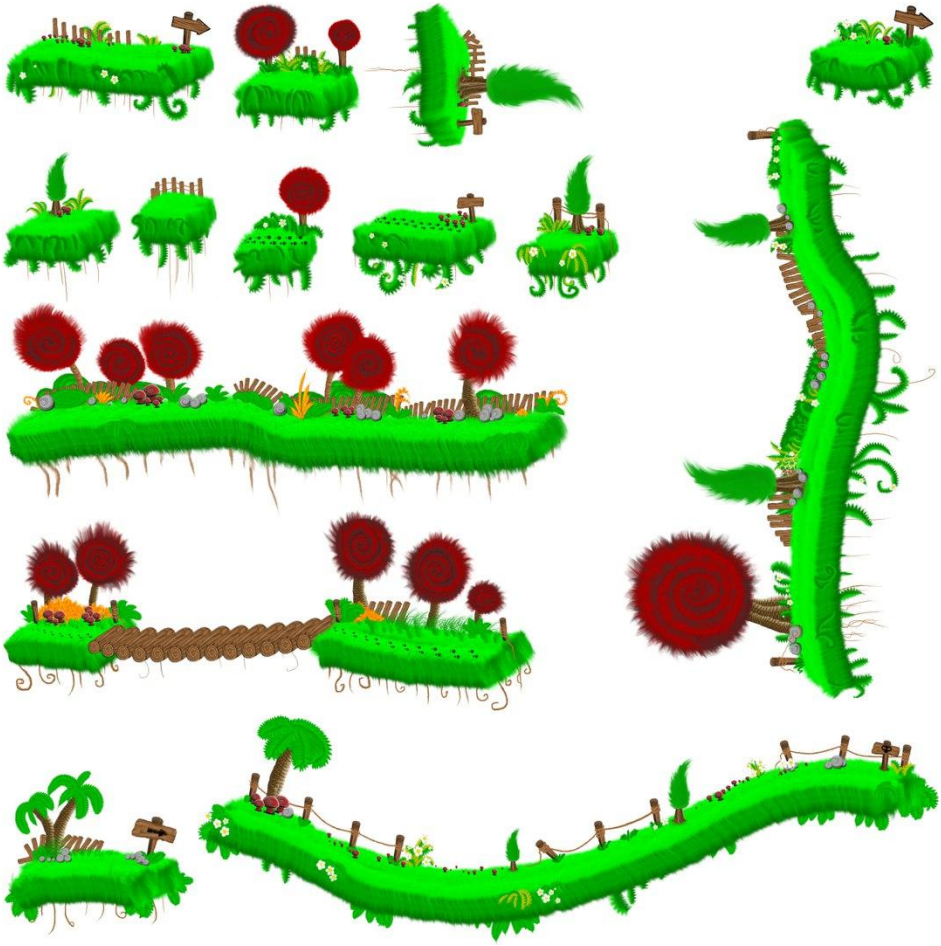


Рисунок 16. Объекты игровой сцены

Описание игры: игра представляет собой платформер и состоит из нескольких уровней, которые открываются по мере прохождения игры. Суть игры заключается в поиске утерянной шляпы. Уровень считается пройденным, если собраны все предметы, находящиеся на уровне. Начиная с первого уровня, в игре появляются враги, которые преграждают путь и атакуют героя. Управление на мобильных устройствах осуществляется при помощи сенсорного экрана, в то время как на персональных компьютерах процесс игры регулируется при помощи мышки и стрелок на клавиатуре. Игра предназначена для любой целевой аудитории и не носит тяжелой смысловой нагрузки. Она используется в целях развлечения пользователя.

При запуске игры открывается главное меню. В меню можно начать новую игру, загрузить сохраненную игру, настроить звуковое сопровождение, открыть информацию об игре или о разработчике.

Цель игры: пройти все уровни, чтобы найти потерянную шляпу.

Структуру приложения можно представить следующим образом:

- Animations;
- Materials;
- Prefabs;
- Scenes;
- Scripts;
- Textures.

Каждый элемент списка представляет собой каталог, который содержит соответствующие материалы.

Animations – каталог с анимациями, которые используются для анимации меню, персонажей и игрового окружения.

Materials – каталог содержит физические материалы, из которых состоят объекты. Материалы позволяют моделировать физические процессы, такие как отталкивание или воздействие силы.

Prefabs – каталог с объектами, из которых строится каждый уровень.

Scenes – в этом каталоге расположены все сцены приложения.

Scripts – каталог с исходными кодами (скриптами).

Textures – каталог, содержащий все текстуры, которые используют объекты приложения.

Основные сцены приложения:

- mainMenu – сцена для отображения главного меню и меню настроек;
- aboutGame – сцена для отображения информации об игре;
- levels – сцена выбора уровня;
- aboutAuthor – сцена для отображения информации о разработчике;

- level1–level3 – игровые сцены с уровнем 1-3;

mainMenu – главная сцена, которая загружается при запуске приложения. На ней расположено главное меню, состоящее из следующего списка:

- новая игра;
- загрузить игру;
- настройки;
- об игре;
- о разработчике;
- выход.

aboutGame – сцена с описанием правил игры. На сцене расположен текст, описывающий цель игры и управление.

levels – сцена с меню выбора пройденных уровней.

aboutAuthor – на сцене расположен текст, указывающий разработчика, правообладателя и связь с разработчиком.

Level1 – level3 – сцены с игровыми уровнями. На каждой игровой сцене обязательно присутствуют: платфомы, персонаж, фон. Платформы необходимы для передвижения по ним персонажа. Однако между некоторыми платформами существует пустое пространство, при падении в которое прохождение уровня начинается заново. Также существуют препятствия, среди которых отрицательные персонажи, атакующие главного героя. В конце каждого уровня находится «портал», позволяющий перейти на новый уровень. В правом верхнем углу находится кнопка возврата в главное меню.

Для передвижения персонажа и его взаимодействия с игровыми объектами созданы скрипты написанные на языке C# [19]:

- CameraMovement – скрипт, отвечающий за передвижение камеры, которая следует за персонажем;
- LevelManager – скрипт, отвечающий за управление уровнем (переключение уровней, включение и выключение объектов);

- PlayerController – скрипт, отвечающий за управление персонажем;
- EnemyController – скрипт, описывающий логику поведения врагов;
- AudioLvL – скрипт для управления громкостью звуков и музыки;
- AudioMute – скрипт для включения и отключения звуков и музыки;
- MusicManager – скрипт для воспроизведения музыки.

Наиболее интересными и значимыми являются код для передвижения камеры и для передвижения персонажа.

CameraMovement.cs

```
using UnityEngine;
using System.Collections;
//движение камеры за целью
public class CameraMovement : MonoBehaviour {
 public Transform target; //цель, за которой будет двигаться камера
 public float smoothing = 3f; //коэффициент нелинейности движения, чем меньше, тем сильнее камера отстает от цели
        // Use this for initialization
        void Start () {
        }
        // Update is called once per frame
        void Update () {
 Vector3 destPoint = new Vector3(target.position.x, target.position.y, -10); //определяем точку, в которую нужно двигать камеру
 transform.position = Vector3.Lerp(transform.position, destPoint, Time.deltaTime * smoothing); //двигаем камеру
        }
}
```

PlayerController.cs

```
using UnityEngine;
using System.Collections;
```

```
//управление персонажем игрока
public class PlayerController : MonoBehaviour {
 public float maxSpeed = 2f; //скорость движения
 public float jumpForce = 7.5f; //сила прыжка
 public bool isGrounded = false; //провка, на земле ли персонаж
 public bool left = false; //движение влево
 public bool right = false; //движение вправо
 public bool isFacingRight = true; //куда смотрит персонаж, для поворота спрайта
 private Animator anim; //управление анимациями персонажа
 private Rigidbody2D rigidbody2d; //здесь хранятся физические свойства персонажа
 public Transform groundCheck;//объект для проверки в прыжке персонаж или на земле
 public float groundRadius = 0.2f; //радиус проверки
 public float t;
 public LayerMask whatIsGround; //то, что считаем землей
 public bool fall = false; //истина, если персонаж падает (не прыжок)
 public bool sit = false; //персонаж сидит
 public bool jump = false; //персонаж прыгает
 public AudioClip[] ac; //набор звуков
 public AudioSource asrc; //компонент управления звуком
 public LevelManager lvlm ;//ссылка на менеджер уровня
 public int mute;
 public float lvl; //уровень громкости
 // Use this for initialization
 void Start () { //инициализация начальных значений
 rigidbody2d = gameObject.GetComponent<Rigidbody2D>();
 rigidbody2d.gravityScale = 1; //действие гравитации
```

```
anim = GetComponent<Animator>(); //компонент управления анимацией
asrc = GetComponent<AudioSource>(); //компонент управления звуком
t = 0; //счетчик
mute = PlayerPrefs.GetInt("SMute"); //загружаем значение mute
lvl = PlayerPrefs.GetFloat("SLVL"); //загружаем значение уровня громкости
lvlm.ShowUI(1); //отображение интерфейса управления
}
public void Jump() //метод прыжка
{
if (!jump)
{
asrc.clip = ac[1]; //выбор нужного звука
asrc.Play(); //воспроизведение звука
jump = true; //персонаж в прыжке
rigidbody2d.gravityScale = 1;
isGrounded = false; //оторвался от земли
rigidbody2d.AddForce (new Vector3 (0, jumpForce, 0), ForceMode2D.Impulse); //придаем персонажу импульс вверх
anim.SetBool("Ground", isGrounded); //включаем анимацию прыжка
}
}
public void LeftOn() //идти влево
{
left = true;
}
public void LeftOff() //прекратить идти влево
{
left = false;
```

```
}
public void Sit() //идти вправо
{
if (!sit)
{
sit = true;
anim.SetBool("Sit", sit); //включение нужной анимации
}
}
public void SitOff() //приседание
{
if (sit)
{
sit = false;
anim.SetBool("Sit", sit); //включение нужной анимации
}
}
public void RightOn() //идти вправо
{
right = true;
}
public void RightOff() //прекратить идти вправо
{
right = false;
}

void Update () { //метод вызывается перед каждым обновлением кадра на экране
```

```
fall = !Physics2D.OverlapCircle(groundCheck.position, groundRadius, whatIsGround); //проверка падения
//устанавливает в аниматоре значение скорости взлета/падения, если персонаж не на земле
if (!isGrounded)
{
anim.SetFloat("vSpeed", rigidbody2d.velocity.y);
}
if (!left && !right && isGrounded) //если персонаж не идет и стоит на земле
{
anim.SetFloat("Speed",0);
rigidbody2d.velocity = Vector3.zero; //то убирается ускорение (нужно избежать эфекта скольжения, как на льду)
}
//при нажатии или отпускании клавиш на клавиатуре:
if (Input.GetKeyDown(KeyCode.W))
{
if (!sit) Jump(); //прыжок
}
if (Input.GetKeyDown(KeyCode.S))
{
Sit(); //присесть
}
if (Input.GetKeyUp(KeyCode.S))
{
SitOff(); //встать
}
if (Input.GetKeyDown(KeyCode.A))
```

```
        {
        left = true; //идти влево
        }
        if (Input.GetKeyDown(KeyCode.D))
        {
        right = true; //идти вправо
        }
        if (Input.GetKeyUp(KeyCode.A))
        {
        left = false; //прекратить идти влево
        }
        if (Input.GetKeyUp(KeyCode.D))
        {
        right = false; //прекратить идти вправо
        }
        if (left) //если включено, идти влево
        {
        if (isFacingRight)
        //отражаем персонажа влево
        Flip();
        if(!sit) {
        anim.SetFloat("Speed", 1); //включение нужной анимации
        transform.Translate(new Vector2(-Time.deltaTime * maxSpeed, 0f));
//изменение позиции персонажа
        }
        }
        if (right)
        {
        if (!isFacingRight)
```

```
        //отражаем персонажа вправо
        Flip();
        if (!sit)
        { anim.SetFloat("Speed", 1);
        transform.Translate(new Vector2(Time.deltaTime * maxSpeed,0f));
        }
        }
        }
        void FixedUpdate()
        {
        t += Time.deltaTime;
        if ((t > 0.2f)&&(!isGrounded)) //если персонаж упал
        { anim.SetBool("Ground", false); //включение нужной анимации
        jump = true;
        }
        mute = PlayerPrefs.GetInt("SMute"); //постоянное считывание громкости и
mute
        lvl = PlayerPrefs.GetFloat("SLVL");
        asrc.volume = lvl / 100;
        asrc.mute = mute == 1;
        }
        void OnCollisionEnter2D(Collision2D coll)
        {
        if ((coll.gameObject.tag == "Ground") ||(coll.gameObject.tag == "Golem"))
//если столкновение с землей или големом
        {
        isGrounded = true; //персонаж стоит на земле
        jump = false; //не прыгает
        rigidbody2d.gravityScale = 1f;
```

```
anim.SetBool("Ground", isGrounded); //включение нужной анимации
}
if ((coll.gameObject.tag == "Enemy")||(coll.gameObject.tag == "Zombie") || (coll.gameObject.tag == "Vampire") || (coll.gameObject.tag == "Skeleton")) //если столкновение с врагом
{
PLayDeath(); //звук смерти
lvlm.ShowUI(2); //отображение нужного интерфейса
sit = true;
anim.SetBool("Sit", sit); //воспроизведение нужной анимации
}
if (coll.gameObject.tag == "Hat") //столкновение со шляпой-порталом точно так же, как и с землей
{
isGrounded = true;
jump = false;
rigidbody2d.gravityScale = 10;
anim.SetBool("Ground", isGrounded);
}
}
void OnCollisionExit2D(Collision2D coll) //столкновение исчезло
{
if ((coll.gameObject.tag == "Ground") || (coll.gameObject.tag == "Golem")) //если персонаж не сталкивается с землей или големом
{
rigidbody2d.gravityScale = 1f;
isGrounded = false; //значит произошел прыжок или падение
t = 0;
}
```

```
}
void OnTriggerEnter2D(Collider2D coll) //пересечение с триггером
{
if (coll.gameObject.tag == "DeathTrigger") //если триггер "смерти"
{
PLayDeathScream(); //воспроизводится нужный звук
gameObject.GetComponent<SpriteRenderer>().enabled = false;
lvlm.ShowUI(2); //показывается нужный интерфейс
}
}
private void Flip()
{
//меняем направление движения персонажа
isFacingRight = !isFacingRight;
//получаем размеры персонажа
Vector3 theScale = transform.localScale;
//зеркально отражаем персонажа по оси X
theScale.x *= -1;
//задаем новый размер персонажа, равный старому, но зеркально отраженный
transform.localScale = theScale;
}
public void PLayDeath() //звук смерти от врага
{
asrc.clip = ac[0];
asrc.Play();
}
public void PLayDeathScream() //звук падения
{
```

```
asrc.clip = ac[2];
asrc.Play();
}
}
```

2.3 Результаты апробации

Для реализации апробации было принято решение опубликовать созданное приложение в Google Play. Перед публикацией было проведено тестирование приложения студентами Уральского государственного педагогического университета Института математики информатики и информационных технологий. Для выявления ошибок была создана анкета с вопросами, которая заполнялась после тестирования приложения.

Анкета 1.

Результаты тестирования приложения

1. Укажите модель телефона.

   _____

2. Укажите версию операционной системы Android.
   a) Android 4.2 и ниже;
   b) Android 4.3;
   c) Android 4.4;
   d) Android 5.0;
   e) Android 5.1;
   f) Android 6.0.
3. Укажите разрешение экрана устройства, на которое было установлено приложение.

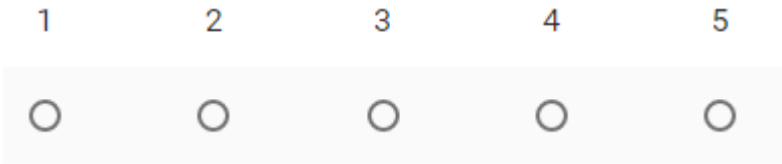
   _____

4. Были проблемы при установке приложения?
   a) Да
   b) Нет

5. Были ошибки в процессе игры, если да, то какие (фон не перекрывал экран целиком, падение сквозь платформы, задержки, аварийное завершение приложения и т.д.)?

   _____

6. Удобный интерфейс?
   a) Да
   b) Нет
7. В игре спрятано "Пасхальное яйцо". Вам удалось его найти?
   a) Да
   b) Нет
8. Понравилась ли игра?
   a) Да
   b) Нет
   c) Затрудняюсь ответить
9. Оцените приложение по шкале от 1 до 5.



10. Пожелания для улучшения игрового приложения.

    _____

Результаты тестирования показали следующее:

1. Игра была установлена и работала под разными версиями операционной системы Android.
2. Элементы интерфейса отражаются без критичных искажений на экране любого разрешения.
3. Проблем с установкой не возникло.
4. Критичных ошибок в процессе игры не было.
5. Интерфейс посчитали удобным 100% тестирующих.
6. Найти "Пасхальное яйцо" удалось 41,% тестирующих.

7. Средняя оценка приложения составила 4,7.

Более подробные результаты приведены в Приложении 2.

Также была подана заявка на регистрацию результата интеллектуальной деятельности. После регистрации созданное приложение будет опубликовано в Google Play.

Для публикации приложения в Google Play необходимо иметь аккаунт разработчика [18]. Регистрация аккаунта осуществляется в 4 этапа:

1. *Вход.* Необходимо войт под уже существующим аккаунтом Google либо создать новую учетную запись.
2. *Принятие соглашения*. Далее будет предоставлено соглашение о распространении программного обеспечения через Google Play, с которым необходимо ознакомиться, после чего принять его условия.
3. *Оплата*. Необходимо оплатить регистрационный сбор, который составляет $25 и взимается один раз.
4. *Ввод данных*. Нужно добавить сведения о разработчике, которые будут видны пользователям на страницах опубликованных приложений.

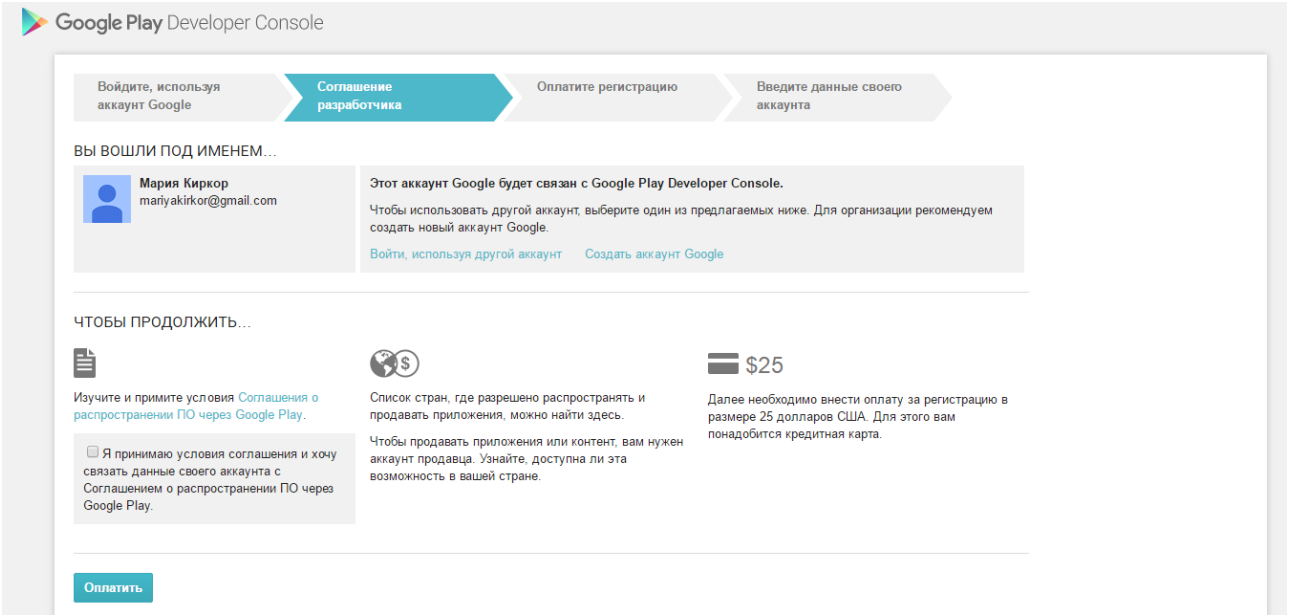


Рисунок 17. Форма регистрации аккаунта разработчика

Для непосредственной публикации приложения необходимо правильное его оформление, которое подразумевает под собой:

1. *Сертификат.* Для приложения необходимо создать сертификат, который позволит идентифицировать разработчика как автора и избежать конфликта имен, если кто-то попытается выложить приложение с таким же именем (под именем подразумевается полное название пакета). При запуске приложения на эмуляторе среда разработки автоматически подписывает программу отладочным сертификатом, однако для публикации необходим уникальный сертификат, по вышеописанным причинам.
2. *Загрузка.* В консоли разработчика необходимо добавить приложение, то есть .APK файл, выбрать язык в открывшемся меню и загрузить рабочую альфа- или бета-версию файла.
3. *Публикация.* Выбрав приложение в консоли разработчика Google Play, можно будет отслеживать статус публикации, чтобы определить доступно ли приложение пользователю:
    - проект – приложение ещё не опубликовано в Google Play;
    - готовится к публикации – приложение обрабатывается;
    - опубликовано – приложение опубликовано и доступно в Google Play;
    - отклонено – приложение не опубликовано из-за нарушения правил Google Play. В данном случае необходимо устранить нарушения и загрузить приложение ещё раз;
    - заблокировано – распространение приложения приостановлено из-за нарушения Правил в отношении контента или Соглашения о распространении программных продуктов. Уведомление об этом будет отправлено на электронную почту владельца аккаунта.
4. *Тестирование.* В аккаунте разработчика доступны альфа- и бета-тестирование приложения, однако если приложение еще не опубликовано (находится в статусах: проект, готовится к публикации, отклонено, заблокировано), то ссылка для тестирования будет недоступна. Тестирование бывает 4 видов:

- Закрытое бета-тестирование – необходимо составить список участников закрытого бета-тестирования, используя адреса электронной почты.
- Открытое бета-тестирование – необходимость в создании каких-либо групп отсутствует, нужно лишь указывать максимальное количество участников.
- Закрытое бета-тестирование для группы Google или сообщества Google+ – необходимо указать адрес электронной почты группы Google или URL сообщества Google+;
- Тестирование игровых сервисов Google Play – если разработчик использует игровые сервисы Google Play, то можено работать с одними и теми же группами альфа- и бета-тестировщиков для проверки APK-файлов и игровых сервисов.

Заключение

В результате исследования был получен материал, анализ которого позволил заключить, что операционная система Android на сегодняшний день является наиболее востребованной, как у разработчиков, так и у конечного пользователя. Также было доказано, что Unity3D обладает широким набором ресурсов позволяющих упростить создание игровых приложений.

В ходе исследования:

1. Проанализированы особенности процесса запуска и исполнения приложений операционной системой Android при помощи виртуальной машины Dalvik и среды выполнения Android-приложений ART;
2. Произведено сравнение мобильных платформ, таких как: Windows Phone, iOS и Android. Сравнение показало, что операционная система Android является наиболее востребованной, гибкой, так как имеет открытый исходный код, а также имеет самую низкую стоимость для приобретения аккаунта разработчика.
3. Рассмотрены особенности программирования под операционную систему Android, которые необходимо знать при создании приложений, такие как:
    - разрешение экрана – может варьироваться от 320x240 пикселей до 1920x1080 и выше;
    - аппаратные конфигурации – у разных устройств, разная производительность;
    - разнообразие программного обеспечения – из-за открытого исходного кода производители устройств могут изменять готовую систему по своему усмотрению;
    - графика – необходимо использовать графику HD формата, чтобы она могла адаптироваться под любое разрешение экрана.
4. Рассмотрены особенности работы операционной системы Android:

- права доступа – все приложения пользуются равными правами доступа к возможностям мобильного устройства;
- установка приложений – для установки приложения необходимо в 2-4 раза больше места, чем размер самого приложения;
- реализация файловой системы – чем меньше места на SD-карте, тем больше времени занимают операции с файлами.

5. Рассмотрены инструменты поддерживающие разработку для операционной системы Android, такие как: Qt, Construct 2, Game Maker: Studio и Unity3D. После тщательного анализа инструментов для создания игрового приложения был выбран Unity3D;
6. Произведен анализ инструмента Unity3D, а именно: рассмотрены работа со сценами, компонентами, игровыми объектами, программирование скриптов на языке C#, графические, физические и аудио возможности Unity3D, выделены его особенности;
7. Разработано игровое приложение «Потерянная шляпа» средствами Unity3D для операционной системы Android;
8. Проведена апробация разработанного игрового приложения в форме тестирования и регистрации результата интеллектуальной деятельности с последующей публикацией в сервисе операционной системы Android Google Play.

Библиографический список

1. Голощапов А.Л. Google Android программирование для мобильных устройств. - СПб: 2011.
2. Майер Р. Программирование приложений для планшетных компьютеров и смартфонов. - М.: Эксмо, 2011.
3. Хашими С., Коматинени С., Маклин Д. Разработка приложений для Android. - М.: Питер, 2011.
4. Цехнер М. Программирование игр для Android . Питер, 2013.
5. Goldstone W. Unity3D Game Development Essentials. 2009.
6. Нахавандипур В. iOS. Разработка приложений для iPhone, iPad и iPod. - СПб: Питер, 2013. - 864 с.
7. Киркор М.А, Газейкина А.И. Использование инструмента Unity3D для разработки игровых приложений под управлением операционной системы Android / Актуальные вопросы преподавания математики, информатики и информационных технологий: межвузовский сборник научных работ. - Екатеринбург: Урал. гос. пед. ун-т. - 2015. - 245 с.
8. Киркор М.А, Газейкина А.И. Технология разработки игровых приложений под управлением операционной системы Android с использованием инструмента Unity3D/ Актуальные вопросы преподавания математики, информатики и информационных технологий: межвузовский сборник научных работ. - Екатеринбург: Урал. гос. пед. ун-т. - 2016
9. Официальный сайт Unity3D URL: http://Unity3D3d.com
10. Официальный сайт Qt URL: http://www.qt.io/ru/developers-2/
11. Официальный сайт Construct 2 URL: https://www.scirra.com
12. Официальный сайт GameMaker URL: https://www.yoyogames.com/studio
13. Официальный сайт Android URL: http://www.android.com
14. Официальный сайт Microsoft URL: https://msdn.microsoft.com

15. Статистика мобильных операционных систем за 2-й квартал 2015 года http://www.oszone.net/27945/The_state_of_the_smartphone_market_Q2_2015_Gartner
16. Официальный сайт сети разработчиков Microsoft URL: https://msdn.microsoft.com/ru-ru
17. Технология разработки программного обеспечения URL: http://www.tehprog.ru/index.php_page=lecture12.html
18. Справочный центр – Google Play Developer Console URL: https://support.google.com/googleplay/android-developer
19. Андерс Хейлсберг Язык программирования C#. Классика Computers Science. - 4 изд. - СПб: Питер, 2012. - 784 с.

Приложения

Приложение 1

Приобретение аккаунта разработчика

Windows Phone Store [14]. Для того чтобы опубликовать свое приложение в Windows Phone Store необходимо приобрести аккаунт разработчика стоимостью $49/год для частного лица и $99/год для юридического. Однако для студентов, аспирантов, школьников старше 12 лет и преподавателей действует программа DreamSpark, которая позволяет бесплатно размещать приложения в App Hub (специальный ресурс для размещения приложений и проверка их текущего статуса, включая статистику загрузок) для того, чтобы в дальнейшем они попали в Windows Phone Store.

App Store. Приобретение аккаунта разработчика iOS оценивается в $99/год. Любое приложение может быть опубликовано в App Store как в бесплатном доступе, так и в платном. Цена, устанавливаемая на приложение, должна находиться в интервале от $0,99 до $999,99. Также стоит отметить, что Apple забирает себе 30% прибыли от опубликованного приложения.

Google Play. Стоимость аккаунта разработчика, позволяющего публиковать приложения, составляет $25. Однако публикация платных приложений доступна разработчикам не всех стран (Россия и США входят в список стран, которые могут распространять платные приложения). От опубликованного платного приложения разработчик получает 70% прибыли в связи с тем, что Google также как и Apple берет комиссию в размере 30% от приносимого приложением дохода.

Приложение 2

Результаты анкетирования

Рисунок 18. Анкетирование. Вопрос 2

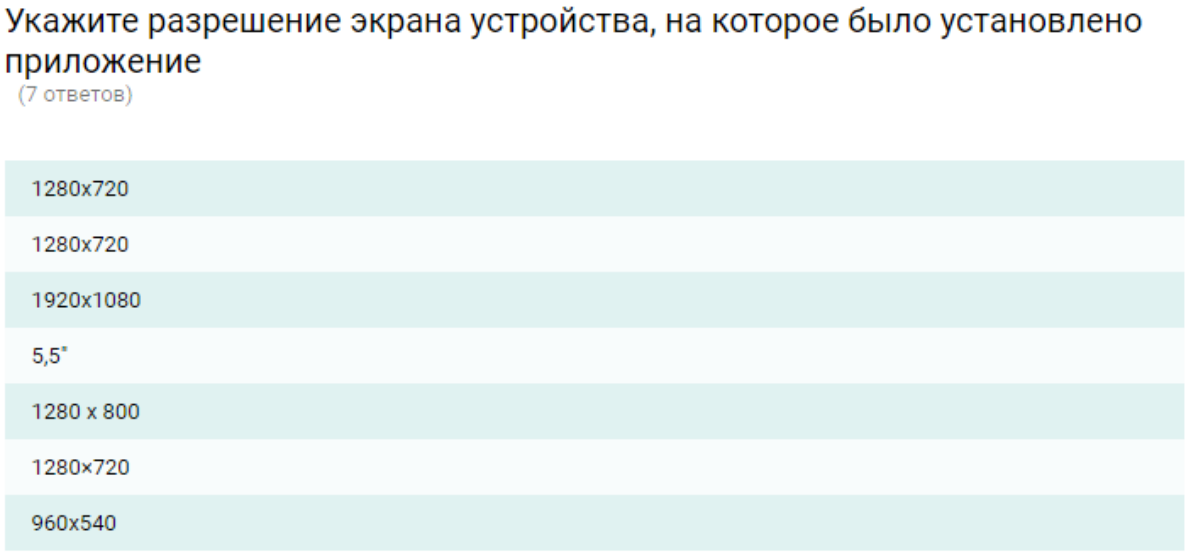


Рисунок 19. Анкетирование. Вопрос 3

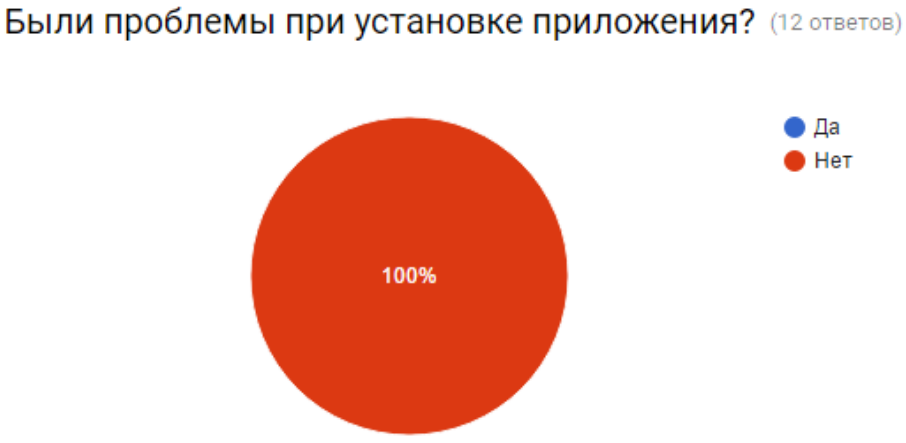


Рисунок 20. Анкетирование. Вопрос 4

Были ошибки в процессе игры, если да, то какие (фон не перекрывал экран целиком, падение сквозь платформы, задержки, аварийное завершение приложения и т.д.)?

(12 ответов)

Нет
Нет
Нет
Нет
Нет
Всё было нормально.
Падение сквозь шляпу было
Ошибок не было
нет
не было
Несколько раз застревал в траве на краю, но прыжок все исправлял, так что это не критично.
Падение за пределы

Рисунок 21. Анкетирование. Вопрос 5

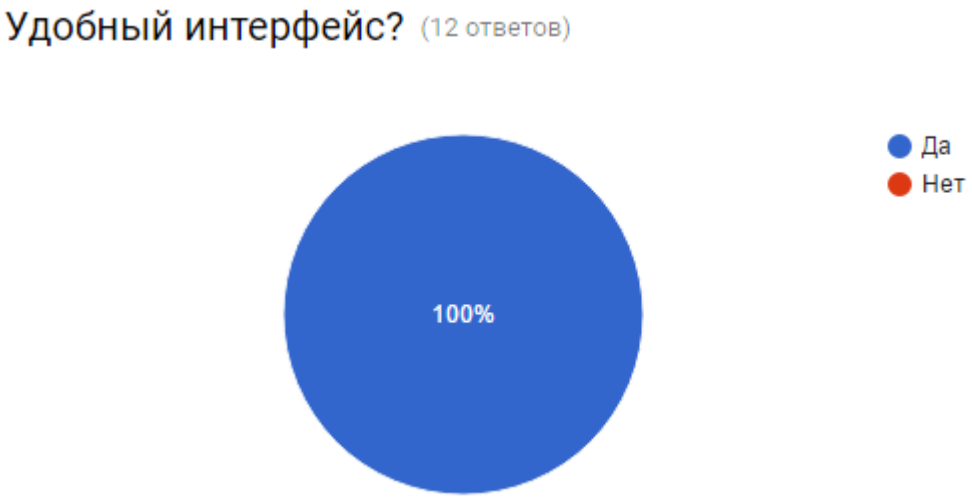


Рисунок 22. Анкетирование. Вопрос 6

Рисунок 23. Анкетирование. Вопрос 7

Рисунок 24. Анкетирование. Вопрос 8

Рисунок 25. Анкетирование. Вопрос 9